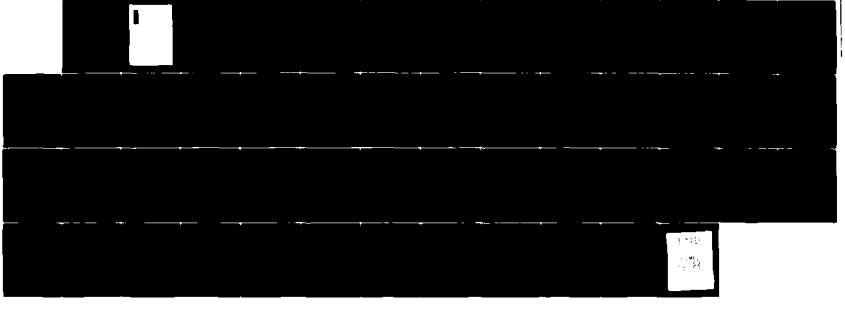
AD-A195 247 THE REDUCTION OF COMPONENT RELIABILITY DISTRIBUTIONS TO A SYSTEM RELIABIL.. (U) CITY UNIV LONDON (ENGLAND) DEPT OF MATHEMATICS A WINTERBOTTOM ET AL. DEC 87 1/1

UNCLASSIFIED DAJA45-86-C-0026 F/G 12/3 NL



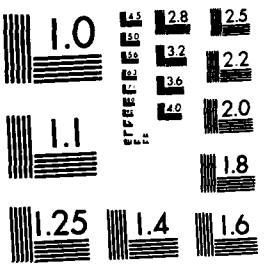


TABLE OF CONTENTS

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

AD-A195 247

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* The Reduction of Component Reliability Distributions to a System Reliability Distribution.		5. TYPE OF REPORT & PERIOD COVERED December 1987 Final Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Alan Winterbottom and John Snell		8. CONTRACT OR GRANT NUMBER(s) DAJA45-86-C-0026
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Mathematics, The City University, Northampton Square, London EC1V OHB, England.		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS USARDSG-UK PO Box 65 FPO NY NY 09510		12. REPORT DATE December 1987 13. NUMBER OF PAGES 48
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Bayes, lower limits, system reliability, component test data, reduction methods.

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

An analagous Bayesian approach is given to the classical methodology for calculating lower bounds on System Reliability. Pass/fail test data and exponential times to failure are considered.

Computer programs are developed to implement the reduction methods.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

SUMMARY

In this ~~final Technical~~ Report an analagous Bayesian approach is given to the classical methodology for calculating lower bounds on system reliability as formulated in the Maximus Handbook of February 1980.

For ease of comparison the Maximus format is adhered to as far as possible. All theoretical work is given in appendices, as is the description of the computer programs that were developed to facilitate calculations.

In addition to the pass/fail test data considered in Maximus the case of exponential times to failure of components is treated.

Key Words:- Bayes, lower limits, system reliability, component test data, reduction methods.

1. INTRODUCTION

It is pertinent to refer to remarks made in the Introduction to the Maximus Handbook. The first remark recognises the need for continued research and analysis and that the methodology of Maximus should be considered as interim in nature. The second remark states that the limitation is to Classical Statistical Theory and that extension of the scope of the work to cover Bayesian methods is deferred.

The present report addresses these two points. Of principal interest is the Bayesian analog of the Classical Maximus methodology. However during the course of development, it became necessary to take a fresh look at the latter and one consequence is that we suggest improvements to the Lindstrom and Madden method for series systems and also for the case of repeated components.

It is not surprising that there should be a link between Classical and Bayesian approaches to reducing component information to 'equivalent' system information. This is due to the well known relationship between partial Binomial sums and the Incomplete Beta function (Chapter 2).

The Bayesian Statistician expresses uncertainty about the unknown component reliabilities through prior probability distributions. For any given component the test data is used to construct a likelihood function which is then combined with the prior probability distribution, using Bayes' theorem, to give a

posterior distribution of component reliability. The system reliability is then the appropriate function (the system reliability function) of component reliabilities. The determination of the exact system reliability distribution is usually formidably difficult, hence the search for simple but good approximate methods, including the reduction methods developed herein.

The intepretation of interval estimate is different using Classical and Bayes approaches. For example, lower 90% limits have the following interpretations.

(i) Classical: There is at least a 0.90 probability that the lower confidence limit (a random variable) will take on a value lower than the fixed but unknown system reliability.

(ii) Bayesian: There is at least a 0.90 probability that the unknown system reliability (a random variable) is greater than the lower limit.

When random variables and parameters are continuous the phrase 'at least' can be omitted in (i) and also in (ii) for continuously distributed parameters.

Classical limits are referred to as confidence limits. Sometimes Bayesian limits are referred to as Bayesian confidence limits but we shall reserve the use of the word confidence for the classical case and simply use the term Bayesian limits. These are appropriate percentage points of the posterior distribution of reliability. The terms credible intervals and credible limits are increasingly used in the Bayesian context.

Like Maximus we restrict our attention to series/parallel systems. For other structures such as k out of n or quorum structures the methods advocated in DAJA37-82-C-0736 can be used. These methods, based on asymptotic expansions, can also be used to provide a useful check on the accuracy of the reduction methods herein. Exact limits can be obtained at the expense of sometime extensive computer simulation and such simulation was used via the computer programs described in the appendices.

Finally we stress that the same provisos on the models and test conditions obtain as for Maximus.

2. CALCULATING BOUNDS FOR A SINGLE COMPONENT

Suppose that n components of a certain type are tested and that s prove to be reliable. If the true reliability (probability of success) is p then the probability of the above outcome is

$$\binom{n}{s} p^s (1 - p)^{n-s} .$$

Let the uncertainty in p be described by a Beta random variable $\tilde{p}$ with probability density function

$$f(p) = p^{\alpha_0 - 1} (1 - p)^{\beta_0 - 1} / B(\alpha_0, \beta_0), \quad 0 < p < 1,$$

where

$$B(\alpha_0, \beta_0) = \int_0^1 p^{\alpha_0 - 1} (1 - p)^{\beta_0 - 1} \, dp .$$

This distribution is called the prior distribution. Using Bayes' theorem the posterior (after tests) distribution is

$$f(p) = p^{\alpha - 1} (1 - p)^{\beta - 1} / B(\alpha, \beta) ,$$

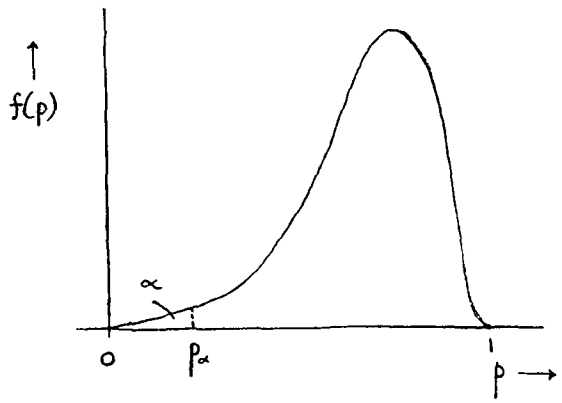
where $\alpha = \alpha_0 + s$ and $\beta = \beta_0 + n - s$.

Since the prior and posterior are both Beta, the Beta distribution is said to be conjugate for pass/fail testing. It is a flexible (two parameter) distribution for expressing uncertainty about component reliability prior to testing. Notice that the prior parameter α_0 is increased by the number of successes and

β_0 is increased by the number of failures. Thus it can be helpful to relate α to successes and β to failures as in Appendix A.3.

Let p_α be such that $\text{Prob}(\tilde{p} \leq p_\alpha) = \alpha$, i.e. $\text{Prob}(\tilde{p} > p_\alpha) = 1 - \alpha$. Then p_α is a lower $100(1 - \alpha)\%$ Bayesian limit for reliability; see Fig. 1.

Figure 1



Often a uniform prior $(\alpha_0 = \beta_0 = 1)$ is used to express vague prior knowledge, whence the posterior probability density function becomes

$$f(p) = p^s(1 - p)^{n-s}/B(s + 1,\ n - s + 1) .$$

Percentage points may be obtained by interpolation in tables of the

incomplete beta function such as Table 16 of Biometrika, Vol. 1. Exact limits can be obtained using the computer program described and listed in Appendix B.

Example 1

Let $s = 11$, $n = 12$ with $\alpha_0 = \beta_0 = 1$. The posterior distribution of reliability is then $B(12, 2)$. Since the parameters are integer Table 16 can be used without interpolation. For $\alpha = 0.1$ the required entry is that for which $\nu_2 = 2 \times 12 = 24$ and $\nu_1 = 2 \times 2 = 4$. This is 0.7322, which is the lower 90% limit for reliability.

3. CALCULATING LOWER LIMITS FOR A SERIES SYSTEM OF NONREPEATED INDEPENDENT COMPONENTS

Let there be k components in the series system and suppose that the posterior distributions for reliability are independent Beta distributions $B(\alpha_i, \beta_i)$; $i = 1, 2, \ldots, k$.

The components are numbered $1, 2, \ldots, k$ so that $\alpha_1 + \beta_1 \geq \alpha_2 + \beta_2 \geq \ldots \geq \alpha_k + \beta_k$. The reduction rules are described for $k = 2$ and extend immediately for general k.

Rule (i): If $\alpha_1 > \alpha_2 + \beta_2$ then take $\alpha_S = \alpha_2$ and $\alpha_S + \beta_S = \frac{\alpha_1 + \beta_1}{\alpha_1} \times (\alpha_2 + \beta_2)$. Figure 2 depicts the scaling down procedure.

Figure 2

$$\alpha_1 + \beta_1 \rightarrow \frac{\alpha_1 + \beta_1}{\alpha_1} \times (\alpha_2 + \beta_2) = \alpha_S + \beta_S$$

$$\alpha_1 \searrow \alpha_2 + \beta_2$$

$$\alpha_2 \rightarrow \alpha_2 = \alpha_S$$

Rule (ii): If $\alpha_1 < \alpha_2 + \beta_2$ take $\alpha_S + \beta_S = \alpha_1 + \beta_1$ and $\alpha_S = \frac{\alpha_1 \alpha_2}{\alpha_2 + \beta_2}$. Scaling down is shown in Figure 3.

Figure 3

$$\alpha_1 + \beta_1 \longrightarrow \alpha_1 + \beta_1 = \alpha_S + \beta_S$$

$$\alpha_2 + \beta_2 \longrightarrow \alpha_1$$

$$\alpha_2 \longrightarrow \frac{\alpha_1 \alpha_2}{\alpha_2 + \beta_2} = \alpha_S$$

When $\alpha_1 = \alpha_2 + \beta_2$ then $\alpha_S + \beta_S = \alpha_1 + \beta_1$ and $\alpha_S = \alpha_2$. In this case the reduction is exact (see A.3).

Example 2

Let $k = 3$; $\alpha_1 = 28$, $\beta_1 = 2$; $\alpha_2 = 21$; $\beta_2 = 20$, $\alpha_3 = 20$, $\beta_3 = 2$. Since $\alpha_1 > \alpha_2 + \beta_2$ take $\alpha + \beta = (\alpha_1 + \beta_1)(\alpha_2 + \beta_2)/\alpha_1 = (30 \times 24)/28 = 25.7143$ and $\alpha = \alpha_2 = 21$. Rewrite α as α_2 and β as β_2. Finally, since α_2 (=21) $< \alpha_3 + \beta_3$ (=22), the system $\alpha_S = (21 \times 20)/22 = 19.0909$ and $\beta_S = 6.6234$.

The lower 90% limit for system reliability obtained by simulating 10,000 values from the exact posterior distribution is 0.627. The limit given by the above approximating beta distribution is 0.629.

Exponential Times to Failure

Suppose that a given component has exponential failure rate λ. Then, given λ, the time to failure x has probability density function

$$f(x) = \lambda e^{-\lambda x}, \quad x > 0, \quad \lambda > 0 .$$

Let the uncertainty in λ be described by a gamma distribution with probability density function

$$g(\lambda) = \tau_0(\tau_0\lambda)^{\eta_0 - 1} e^{-\tau_0\lambda}/\Gamma(\eta_0) ,$$

$\tau_0 > 0$, $\eta_0 > 0$. The likelihood function when n components have failed in a total time on test t is

$$\ell = \lambda^n e^{-\lambda t} .$$

The posterior distribution of the failure rate is then

$$g(\lambda) = \tau(\tau\lambda)^{\eta-1} e^{-\tau\lambda}/\Gamma(\eta) ,$$

when $\eta = \eta_0 + n$ and $\tau = \tau_0 + t$. Again we have a flexible prior distribution which is conjugate for exponential testing. Now, without loss of generality, we may take the unit of time to be the mission time whence the reliability of the component is

$$p = e^{-\lambda} .$$

It follows that the distribution of p has probability density function

$$f(p) = \tau^{\eta}(-\ell np)^{\eta-1} p^{\tau-1}/\Gamma(\eta) .$$

This is precisely the distribution of the product of η independent $B(\tau, 1)$ variates. Thus such a component in a structure is equivalent to η independent series components each with a $B(\tau, 1)$ distribution. If η is an integer than the reduction methods given previously for Beta variates give a single approximating Beta

distribution with $\alpha = \tau^{\eta}/(\tau + 1)^{\eta-1}$ and $\beta = \{(\tau + 1)^{\eta} - \tau^{\eta}\}/(\tau + 1)^{\eta-1}$. Even if η is noninteger the above approximation is still very good. Note that in Maximus noninteger sample sizes and successes are common under reduction.

Example 3

Let $\eta = 3$, $\tau = 40$. Then $\alpha = 40^3/41^2 = 38.07$ and $\beta = \{41^3 - 40^3\}/41^2 = 2.93$. The exact lower 90% limit for reliability is $\exp\{-\lambda(.1)\}$, where $\lambda(.1)$ is the 10% point of the distribution of $\tilde{\lambda}$. Since it is well known that $2\tilde{\lambda}\tau$ has a chi-squared distribution with 2η degrees of freedom we have $\lambda(.1) = \chi^2_6(.1)/2 \times 40 = 0.1331$ giving $e^{-\lambda(.1)} = 0.875$. Using the incomplete beta function with $\alpha = 38.07$ and $\beta = 2.93$ we get the corresponding approximate lower 90% limit of 0.864.

4. CALCULATING LOWER LIMITS FOR A PARALLEL SYSTEM OF NONREPEATED INDEPENDENT COMPONENTS

Let there be k components in the parallel system and suppose that the posterior distributions of component reliabilities are, as before, independent $B(\alpha_i, \beta_i)$ distributions; $i = 1, 2, \ldots, k$. The reliability of the parallel system for given component reliabilities $p_1, p_2, \ldots, p_k$ is

$$p_s = 1 - \prod_{i=1}^{k}(1 - p_i)$$

$$= 1 - \prod_{i=1}^{k} q_i, \quad \text{where} \quad q_i = 1 - p_i .$$

Thus $q_s = 1 - p_s = \prod_{i=1}^{k} q_i$.

If $\tilde{p}_i \sim B(\alpha_i, \beta_i)$ then $\tilde{q}_i \sim B(\beta_i, \alpha_i)$ and the results for series systems can be used here by interchange of parameters and of reliabilities by failure probabilities. Also one needs to scale up rather than scale down, as is the case for series sytems. The following example illustrates the procedure.

Example 4 $k = 3$; $\alpha_1 = 6$, $\beta_1 = 2$; $\alpha_2 = 6$, $\beta_1 = 2$; $\alpha_3 = 9$, $\beta_3 = 1$.

Component:	1	2	3
$\alpha + \beta$:	8	8	10
(α) :	(6)	(6)	(9)
β :	2	2	1
		↓	↙
$\frac{8}{2} \times 8 =$		32	10
		(30)	(9)
		2	1

The rule for scaling up when sample sizes are unequal is 'scale so that $\alpha_s + \beta_s$ is a minimum'. Thus we have

32	10
(30)	(9)
2	1

$$\alpha_s + \beta_s = 16 \times 10 = 160$$

$$\alpha_s = (159)$$

$$\beta_s = 1$$

5000 simulated values of system reliability gave a lower 95% limit of 0.975. The lower 95% limit given by the approximating B(159, 1) variate is 0.980.

5. CALCULATING LOWER LIMITS FOR SERIES AND PARALLEL SYSTEMS OF INDEPENDENT REPEATED COMPONENTS

5(a) Series Systems

Suppose that the posterior distribution for the reliability of a component is $B(\alpha, \beta)$ and that k such components are connected serially. The reliability of the series system is p^k, where p is the individual component reliability. A good approximating beta distribution has parameters α_s, β_s given by

$$\alpha_s = \beta\left[\left\{1 - \frac{\alpha(\alpha+1)\ldots(\alpha+k-1)}{(\alpha+\beta)(\alpha+\beta+1)\ldots(\alpha+\beta+k-1}\right\}^{-1} - 1\right],$$

$$\beta_s = \beta.$$

The derivation of these expressions for α_s, β_s is given in A.4.

Of course it is easy to obtain exact lower limits for p^k since they are simply the corresponding limits for p raised to the power k. However, when such a structure is embedded in a more complex arrangement, with components of different types, we do require a reduction to a single beta distribution. The following examples show that the above reduction works very well. It is based on the adapted sequential procedure plus the method of moments - see A.4.

Example 5

$k = 2$; $\alpha = 30$, $\beta = 2$.

$$\alpha_s = 2\left[\left\{1 - \frac{30 \times 31}{32 \times 33}\right\}^{-1} - 1\right] = 14.762$$

$$\beta_s = 2.$$

Thus B(14.762, 2) is the reduced beta variate for p^2. The exact lower 90% limit for p^2 from Table 16 of Biometrika, Vol. 1 is $(0.88023)^2 = 0.7748$. The approximate limit given by the above reduced beta variate is 0.7747, in excellent agreement with the exact result. Maximus advocates equal splitting for repeated components in series. The Lindstrom and Madden method is then applied as if the components were of different types. The reduced beta variate is then B(14.06, 1.94), using the analagous procedures of Chapter 3. The approximate lower 90% limit for p^2 is 0.770, which is not as good as the adapted sequential approach.

Example 6

$K = 4$; $\alpha = 30$, $\beta = 2$

$$\alpha_s = 2\left[\left\{1 - \frac{30 \times 31 \times 32 \times 33}{32 \times 33 \times 34 \times 35}\right\}^{-1} - 1\right] = 7.154,$$

$$\beta_s = 2.$$

The exact lower 90% limit is $(0.88023)^4 = 0.6003$ and the approximate limit is 0.5999, again in excellent agreement. Equal splitting of parameters gives 0.5770.

5(b) Parallel Systems

The formula for repeated components in parallel is given in A.4(ii). We have

$$\alpha_s = \alpha$$

$$\beta_s = \alpha\left[\left\{1 - \frac{\beta(\beta+1)\ldots(\beta+k-1)}{(\alpha+\beta)(\alpha+\beta+1)\ldots(\alpha+\beta+k-1)}\right\}^{-1} - 1\right]$$

Example 7

$k = 4$; $\alpha = 8$, $\beta = 2$

$$\alpha_S = 8$$

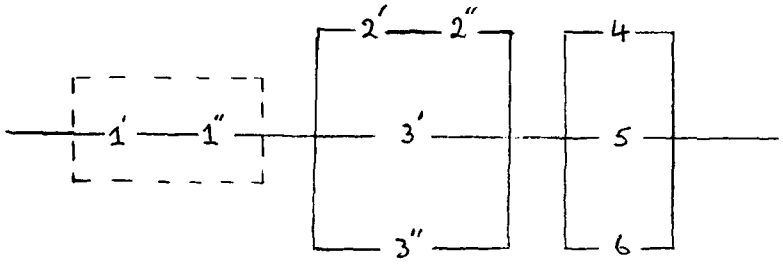
$$\beta_S = 8\left\{1 - \frac{2 \times 3 \times 4 \times 5}{10 \times 11 \times 12 \times 13}\right\}^{-1}$$

Exact: The lower 95% limit for system reliability is 1 - the upper 95% limit for failure probability. For q this limit is 0.4295 and therefore exact lower 95% limit for reliability is $1 - 0.0326 = \underline{0.967}$.

Approximate: Using the approximating $B(8, .0563)$ variate the lower 95% limit is $\underline{0.960}$.

6. TWO COMPLEX STRUCTURES

Structure 1



(1', 1"), (1', 1"), (3', 3") signify replicates of components 1, 2 and 3 respectively.

	Beta Parameters	
Component	α	β
1	29	1
2	10	1
3	9	1
4	8	2
5	8	2
6	8	2

Using the reduction methods described previously

M1 = (1' * 1") $\Rightarrow$ B(14.5, 1); (2' // 2") $\Rightarrow$ B(5, 1); (3' // 3") $\Rightarrow$ B(9, 0.167)

M2 $\Rightarrow$ B(45.83, 0.167), M3 $\Rightarrow$ B(248, 2).

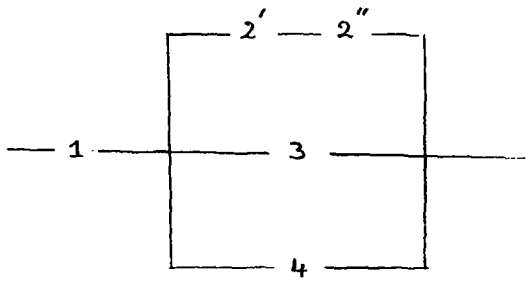
M1 * M2 * M3 $\Rightarrow$ B(14.5, 1.182).

Approximate lower 95% limit = 0.792

Exact lower 95% limit = 0.809.

(Based on 5000 simulated system values).

Structure 2



(2', 2") are replicates of component 2.

	Beta Parameters		Gamma Parameters	
Component	α	β	η	τ
1			2	40
2	9	1		
3	8	2		
4			3	25

Converting the Gamma distributions to Series Beta distributions using the results of Chapter 3 we get for component 1 a B(38.025, 1.985) variate and component 4 reduces to a B(22.114, 2.886) variate. The final system approximating Beta distribution has parameters 38.025 and 2.151. This yields an approximate lower 95% limit of 0.8806.

5000 simulated values of system reliability using the computer program gave a corresponding lower limit of 0.8859.

APPENDIX A

An Alternative to Lindstrom and Madden

The Lindstrom and Madden method is recommended for series systems by Maximus. We first describe an exact sequential procedure for series systems and then show how it can be adapted to deal with arbitrary but fixed test sample sizes. The proof of an interesting property of the adapted procedure is then given. Some comparisons with results given by Lindstrom and Madden and also some other approximate procedures are presented. Finally it is shown how the modified sequential procedure can be converted to the analagous Bayesian procedure given in Ch. 3.

A.1 An Exact Sequential Procedure

Let there be k independent components in series. The component types may be arbitrarily labelled $1, 2, \ldots, k$. Suppose n_1 components of type 1 are tested and that s_1 prove to be reliable. Take the sample size for component 2 to be $n_2 = s_1$ and suppose that s_2 are reliable. Continuing in this way the sample size for the k^{th} component is $n_k = s_{k-1}$ and let s_k of these be reliable. Then in random sampling $\tilde{s}_k$ is Binomially distributed with n_1 trials and parameter $p = \Pi_{i=1}^{k} p_i$. Write

$$\tilde{s}_k \sim \text{Bin}(n_1, p).$$

Proof that $\tilde{s}_k \sim \text{Bin}(n_1, p)$

Let $k = 2$. Given $n_2 = s_1$ the probability generating function for $\tilde{s}_2$ is

$$G_2(z|n_2 = s_1) = (q_2 + p_2 z)^{s_1}, \quad q_2 = 1 - p_2 .$$

The unconditional probability generating function is

$$G_2(z) = \sum_{s_1=0}^{n_1} (q_2 + p_2 z)^{s_1} \binom{n_1}{s_1} p_1^{s_1} (1 - p_1)^{n_1 - s_1}$$

$$= \sum_{s_1=0}^{n} \binom{n_1}{s_1} [(q_2 + p_2 z)p_1]^{s_1} (1 - p_1)^{n_1 - s_1}$$

$$= [(q_2 + p_2 z)p_1 + (1 - p_1)]^{n_1}$$

$$= [(1 - p_1 p_2) + p_1 p_2 z]^{n_1}$$

$$= (q + p)^{n_1}, \quad p = p_1 p_2, \quad q = 1 - p .$$

The result for k components in series follows by induction.

A.2 A Forced Sequential Procedure

Now let $n_1, n_2, \ldots, n_k$ be arbitrary but fixed sample sizes and let $s_1, s_2, \ldots, s_k$ be the corresponding numbers of reliable components obtained in tests. Without loss of generality label the components so that $n_1 \geq n_2 \geq \ldots \geq n_k$. For simplicity let $k = 2$ initially. Two cases need to be considered,

(i) $s_1 \geq n_2$ and (ii) $s_1 < n_2$.

The rule to 'force' the sequential procedure for case (i) is - sample at random and without replacement from the test results for component 1 and stop when n_2 successes have been obtained. The average sample size

to achieve n_2 successes is $n_2 \times (n_1 + 1)/(s_1 + 1)$. This is taken to be the system sample size and the number of reliable systems is taken to be s_2. For case (ii) choose a random sample of size s_1 without replacement from the test results for component 2. The average or expected number of reliable items is $s_1 \times s_2/n_2$ and this is taken to be the number of reliable systems in a sample of size n_1. In both cases we assume that the results are binomial test results and lower confidence limits for system reliability can be obtained from binomial reliability tables, interpolating as necessary for noninteger sample sizes and succcesses (number reliable). For the above procedure both the sample size N and the number reliable in the sample S are random variables connected by the interesting result

$$E(S) = p\,E(N),$$

where $p = p_1 p_2$. The procedure extends easily to general k.

Proof that $E(S) = p\,E(N)$

$$E(N) = n_2 \sum_{s_1=n_2}^{n_1} \frac{\binom{n_1+1}{s_1+1}}{} p(S = s_1) + n_1\, p(S_1 < s_2)$$

$$= n_2 \sum_{s_1=n_2}^{n_1} \frac{\binom{n_1+1}{s_1+1}}{}\binom{n_1}{s_1} p_1^{s_1}(1 - p_1)^{n_1-s_1} + n_1 \sum_{s_1=0}^{n_2-1} \binom{n_1}{s_1} p_1^{s_1}(1-p_1)^{n_1-s_1}.$$

$$E(S) = n_2 p_2\, P(S_1 \geq n_2) + p_2 \sum_{s_1=0}^{n_2-1} s_1\, P(S_1 = s_1)$$

$$= n_2 p_2 \sum_{s_1=n_2}^{n_1} \binom{n_1}{s_1} p_1^{s_1} (1 - p_1)^{n_1-s_1} + p_2 \sum_{s_1=0}^{n_2-1} s_1 \binom{n_1}{s_1} p_1^{s_1}(1-p_1)^{n_1-s_1}$$

Using the combinatorial identity $\binom{x}{y} = \binom{x-1}{y} + \binom{x-1}{y-1}$ we can write

$$\sum_{s_1=n_2}^{n_1} \binom{n_1+1}{s_1+1} p_1^{s_1+1} (1-p_1)^{n_1-s_1} = \sum_{s_1=n_2}^{n_1} \binom{n_1}{s_1} p_1^{s_1} (1-p_1)^{n_1-s_1} - \binom{n_1}{n_2} p_1^{n_2} (1-p_1)^{n_1-n_2+1}.$$

Thus

$$E(N) = \frac{n_2}{p_1} \sum_{s_1=n_2}^{n_1} \binom{n_1}{s_1} p_1^{s_1} (1-p_1)^{n_1-s_1} - n_2 \binom{n_1}{n_2} p_1^{n_2} (1-p_1)^{n_1-n_2+1}$$

$$+ n_1 \sum_{s_1=0}^{n_2-1} \binom{n_1}{s_1} p_1^{s_1} (1-p_1)^{n_1-s_1}.$$

A second application of the combinatorial identity to the last term on the right hand side of $E(N)$ gives

$$E(N) = \frac{n_2}{p_1} \sum_{s_1=n_2}^{n_1} \binom{n_1}{s_1} p_1^{s_1} (1-p_1)^{n_1-s_1} - n_2 \binom{n_1}{n_2} p_1^{n_2-1} (1-p_1)^{n_1-n_2+1}$$

$$+ n_1 \left\{ \sum_{s_1=0}^{n_2-1} \binom{n_2-1}{s_1-1} p_1^{s_1-1} (1-p_1)^{n_1-s_1} + \binom{n_1-1}{n_2-1} p_1^{n_2-1} (1-p_1)^{n_1-n_2+1} \right\}$$

$$= \frac{n_2}{p_1} \sum_{s_1=n_2}^{n_1} \binom{n_1}{s_1} p_1^{s_1} (1-p_1)^{n_1-s_1} + \frac{1}{p_1} \sum_{s_1=0}^{n_2-1} s_1 \binom{n_1}{s_1} p_1^{s_1} (1-p_1)^{n_1-s_1}$$

$$= E(S)/(p_1 p_2)$$

$$E(S) = p\, E(N) .$$

A Comparative Example

$k = 3$; $n_1 = 30$, $s_1 = 28$; $n_2 = 24$, $s_2 = 22$; $n_3 = 20$, $s_1 = 19$. Reducing components 1 and 2 using case (i) gives $n = 25.714$, $s = 22$. Combining these values with those for component 3, again using (i), gives the final values $n = 23.377$, $s = 19$. Linear interpolation in the tables of Cook, Lee and Vanderbeck (1964) gives a lower 90% confidence limit of 0.670 for system reliability.

The Lindstrom and Madden method gives $n = 20$ and $s = 16.256$ yielding a lower 90% confidence limit of 0.654.

There is no unique exact lower limit but other methods suggest that exact procedures would give values between 0.68 and 0.69 - see Example 1 of Winterbottom (1984).

Note that case (ii) is the Lindstrom and Madden procedure. Overall the sequential approach is less conservative than Lindstrom and Madden and merits further study.

A.3 An Analagous Bayesian Sequential Procedure

Consider k independent beta variates $B(\alpha_i, \beta_i)$, $i = 1, 2, \ldots, k$. Suppose that $\alpha_1 + \beta_1 \geq \alpha_2 + \beta_2 \geq \ldots \geq \alpha_k + \beta_k$ and that $\alpha_1 = \alpha_2 + \beta_2$, $\ldots$, $\alpha_{k-1} = \alpha_k + \beta_k$. Then the distribution of the product of these variables is exactly Beta $B(\alpha_k, \sum_{i=1}^{k} \beta_i)$.

Now (see Chapter 2), equating α with s, $\alpha + \beta$ with n and regarding reliability as a random variable with α, β fixed, the

result

$E(S) = p\,E(N) \longrightarrow \alpha = E(P)(\alpha + \beta)$. This suggests the 'forced' sequential procedure of Chapter 3 when the α_i, β_i are arbitrary.

A.4 Repeated Components in Series or Parallel

(i) Series Systems

Let $k = 2$ so that we require an approximating distribution for p^2. Suppose that the distribution of p is $B(\alpha, \beta)$.

Split α, β according to the following sequential scheme

$$\begin{matrix} x & \\ y & = y \\ & z \end{matrix}$$

where $x + y = \alpha + \beta$ and $y + z = \alpha$. Subtraction gives $x - z = \beta$. Knowledge of y is not required since we take $\alpha_s + \beta_s = x$ and $\alpha_s = z$. Thus far x and z are not uniquely determined. By the method of moments equate z/x, the mean of the approximating beta distribution, to the mean of p^2, i.e.

$$z/x = \frac{\alpha(\alpha+1)}{(\alpha+\beta)(\alpha+\beta+1)} .$$

Using $x - z = \beta$, $x = \alpha_s + \beta_s$, $z = \alpha_s$ and solving gives

$$\alpha_s = \beta\left[\left\{1 - \frac{\alpha(\alpha+1)}{(\alpha+\beta)(\alpha+\beta+1)}\right\}^{-1} - 1\right]$$

and $\beta_s = \beta$.

The corresponding expressions for general k are

$$\alpha_s = \beta \left[\left\{ 1 - \frac{\alpha(\alpha+1)\ldots(\alpha+k-1)}{(\alpha+\beta)(\alpha+\beta+1)\ldots(\alpha+\beta+K-1)} \right\}^{-1} - 1 \right]$$

and $\beta_s = \beta$.

(ii) Parallel Systems

Since failure probability $q = 1 - p$ all that is necessary to obtain corresponding results for parallel systems is to interchange α, β in the above results for series systems.

Thus

$$\alpha_s = \alpha$$

amd $$\beta_s = \alpha \left[\left\{ 1 - \frac{\beta(\beta+1)\ldots(\beta+k-1)}{(\alpha+\beta)(\alpha+\beta+1)\ldots(\alpha+\beta+k-1)} \right\}^{-1} - 1 \right].$$

APPENDIX B

B.1 Directed graphs

It is possible to represent a control system by a directed graph where the nodes are the components in the system and where the directed lines represent the flow of control signals.

To cater for branching at the start of a graph it is convenient to introduce an extra node, called the source, before the first node. Thus a source node has no inputs. Similarly, to deal with multiple returns at the end of the graph, an extra node with no outputs, called the sink, is added there. The graph then has a unique entry and a unique exit and the reliability of the whole graph is simply the reliability of the connection between the source and the sink.

In order to describe the graph the nodes are numbered sequentially, starting with zero for the source and reserving the highest number for the sink. It is then possible to store the connection information by forming, for each node, the set of nodes which directly receive its outputs. A convenient representation for this is as a linked list of records called KIN where each record has integer fields FATHER, SIZE and a field SONS, consisting of an integer set, such that the node FATHER has SIZE outputs to nodes called sons, the numerical values of which are held as elements of the set SONS.

If there is a linkage through a succession of nodes from A to B then the corresponding set of node numbers is called a path. Clearly there may be more than one path between two nodes and any node may be a member of several paths. Two paths are said to be independent if their intersection is the null set otherwise they are dependent. Thus dependent paths have nodes in common and independent paths do not. A path which contains both the source and the sink is called a complete path, otherwise it is called an incomplete path. The type of control system which is considered in this report consists only of complete paths.

From the information in the list KIN, all distinct and complete paths may be found. These paths may be manipulated to give the algebraic expression for the overall reliability of the graph and finally this expression may be evaluated numerically.

This process is most suitable for embodiment in a computer program which interrogates the user for the son nodes of each father node, requesting also reliability information for each node and which then proceeds to perform the appropriate manipulations on these data.

B.2 Control system reliability

The algebra for dealing with the reliability of combinations of components in a control system is based upon the two results :-

(a) if nodes A and B are directly connected in series then the reliability of the combination is $r_A r_B$ where r_A and r_B are the reliabilities of A and B,.

(b) if nodes A and B are connected in parallel then the reliability of the combination is $1 - (1 - r_A) (1 - r_B)$.

These results are readily extended to more complex cases of series/parallel connections.

If all control systems were composed of series/parallel combinations of components then the rules (a) and (b) would be sufficient to determine the overall reliability of the system. Now the graph in fig.1 is an example of a control system where it is not possible to discover a pair of nodes which are either in series or in parallel. It follows that some other technique is required to assess the overall reliability of a system in which the nodes are connected in an arbitrary fashion.

The new technique for the analysis of general control systems starts with the determination of all distinct paths from the source. This is achieved by using a linked list of records called ROUTE, each record contains a field PATH consisting of the set of nodes in a path and a field LASTIN which holds the last node that had been added to PATH. Initially ROUTE consists of a single record with PATH = [0] to represent the source node and with LASTIN = 0.

The information in KIN is then accessed to extend the list in ROUTE as follows :-

For each record in ROUTE, consider LASTIN to be a father node having the sons $s_1 .. s_{SIZE}$, then generate and add to ROUTE a total of (SIZE - 1) replicate records to give SIZE copies. If these copies have PATH fields denoted by $p_1 .. p_{SIZE}$ and LASTIN fields $L_1 . L_{SIZE}$, then for k = 1 to SIZE add s_k to p_k and set $L_k = s_k$. If a son to be added is the sink, then the assembly of that particular record is terminated. This process is repeated until each record in ROUTE has been assembled.

All paths in the systems under consideration are complete and it follows that any such system will exhibit overall failure only if every path fails. This means that the paths must be considered, in some sense, to be connected in parallel between the source and the sink. However, this concept needs to be modified by taking account of the dependences between paths.

To describe the algebra required for path manipulation let P be a complete path and let (P) denote the associated reliability expression obtained as the product of the reliabilities of the nodes in P. Then for a control system without branches, there is only one path and (P)

is the system reliability. Also if P and Q are any two independent paths where the nodes in P are connected in series with the nodes of Q then the reliability of the combination is the reliability of the union set for P and Q, thus $\{P\}*\{Q\} = \{P \cup Q\}$. It is convenient also to use the notation $[\ P - Q\ ]$ for the set of elements which are members of P but not members of Q.

Suppose now that S_1 and S_2 are two complete paths and it is required to derive their combined reliability expression. Using T to denote the intersection set of S_1 with S_2, it follows that the nodes in T are linked in series with the parallel combination of the nodes in the sets $[\ S_1 - T\ ]$ and $[\ S_2 - T\ ]$. The reliability of this parallel combination is then

$$\{[\ S_1 - T\ ]\} + \{[\ S_2 - T\ ]\} - \{[\ S_1 - T\ ] \cup [\ S_2 - T\ ]\} .$$

The connection now, in series, of the nodes in T gives

$$\{T\}*\{[\ S_1 - T\ ]\} + \{T\}*\{[\ S_2 - T\ ]\} - \{T\}*\{[\ S_1 - T\ ] \cup [\ S_2 - T\ ]\}$$

which is equivalent to

$$\{T \cup [\ S_1 - T\ ]\} + \{T \cup [\ S_2 - T\ ]\} - \{T \cup ([\ S_1 - T\ ] \cup [\ S_2 - T\ ])\}$$

and this simplifies to

$$\{S_1\} + \{S_2\} - \{S_1 \cup S_2\} .$$

Now the expression $1 - (1 - S_1)*(1 - S_2)$ can be taken to yield $S_1 + S_2 - S_1*S_2$, so that if the operator * is replaced with the union operator and the operators +, - understood to act only on the equivalent reliability expressions then the required reliability expression is obtained.

The reliability for a further complete path S_3 in conjunction with S_1 and S_2 is then given by associating S_3 with the three components above giving successively

$$\{S_1\} + \{S_3\} - \{S_1 \cup S_3\} ,$$

$$\{S_2\} + \{S_3\} - \{S_2 \cup S_3\} ,$$

and $$- \{S_1 \cup S_2\} - \{S_3\} + \{S_1 \cup S_2 \cup S_3\} .$$

The sum of these reliabilities can be derived from the expression $1 - (1 - S_1)*(1 - S_2)*(1 - S_3)$ if, as before, the operator * is replaced by the union operator.

It follows that, with these interpretations, rule (b) can be applied to complete paths to give their combined reliability expression.

The procedure to be adopted for a general control system then becomes :-

(i) find all distinct complete paths ,

(ii) if there are N such paths denoted by S_1, S_2, .. , S_N then form the expression

$$1 - \prod_{k=1,N} (1 - S_k)$$

(iii) evaluate the products in this expression using the given interpretations for the operators ,

(iv) finally, translate the resulting expression in sets into the corresponding reliability expression.

For the system of fig.1 the construction of ROUTE is as follows :-

```
[0]                                  initial path, now add the son of node 0,
[0 1]                                      add the three  sons of node 1,
[0 1 2]  [0 1 3] [0 1 4]                          add the son of node 2,
[0 1 2 5] [0 1 3] [0 1 4]                        add the sons of node 3,
[0 1 2 5] [0 1 3 5] [0 1 3 6] [0 1 4]             add the son of node 4,
[0 1 2 5] [0 1 3 5] [0 1 3 6] [0 1 4 6]           add the son of node 5,
[0 1 2 5 7] [0 1 3 5 7] [0 1 3 6] [0 1 4 6]       add the son of node 6,
[0 1 2 5 7] [0 1 3 5 7] [0 1 3 6 7] [0 1 4 6 7] add the son of node 7
```

to give the four complete paths represented by the sets :-

S_1 = [0 1 2 5 7 8], S_2 = [0 1 3 5 7 8],
S_3 = [0 1 3 6 7 8] and S_4 = [0 1 4 6 7 8].

The reliability expression for the system is now found by evaluating

$(1 - S_1)(1 - S_2) = S_1 \cup S_2 - S_1 - S_2 + 1$

$= [0\ 1\ 2\ 3\ 5\ 7\ 8] - [0\ 1\ 2\ 5\ 7\ 8] - [0\ 1\ 3\ 5\ 7\ 8] + 1,$

then $(1 - S_1)(1 - S_2)(1 - S_3) =$

$1 - [0\ 1\ 2\ 5\ 7\ 8] - [0\ 1\ 3\ 5\ 7\ 8] - [0\ 1\ 3\ 6\ 7\ 8] + [0\ 1\ 2\ 3\ 5\ 7\ 8]$
$+ [0\ 1\ 3\ 5\ 6\ 7\ 8]$.

and finally, $1 - (1 - S_1)(1 - S_2)(1 - S_3)(1 - S_4) =$

$[0\ 1\ 2\ 5\ 7\ 8] + [0\ 1\ 3\ 5\ 7\ 8] + [0\ 1\ 3\ 6\ 7\ 8] + [0\ 1\ 4\ 6\ 7\ 8]$
$- [0\ 1\ 2\ 3\ 5\ 7\ 8] - [0\ 1\ 3\ 4\ 6\ 7\ 8] - [0\ 1\ 3\ 5\ 6\ 7\ 8]$
$- [0\ 1\ 2\ 4\ 5\ 6\ 7\ 8] + [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8].$

The corresponding reliability expression, assuming that the source and sink reliabilities are both unity, is then found to be :-

$$r_1(r_2r_3 + r_3r_5 + r_3r_4 + r_4r_6 - r_2r_3r_5 - r_3r_4r_6 - r_3r_5r_6 - r_2r_4r_5r_6 + r_2r_3r_4r_5r_6)r_7 \quad .$$

B.3 Numerical assessment of reliability

When the nodes in a control system or a program have given numerical values for their reliabilities then it is a simple matter to evaluate the overall reliability by direct substitution in the reliability expression.

However, if each node has an assumed reliability distribution then it should in principle be possible to find the overall reliability distribution as an analytic expression. The problem becomes one of finding the theoretical distribution for a weighted sum of products of variates with known distributions.

A simpler and more practical alternative to this theoretical approach uses a Monte Carlo method. Here the reliability expression is evaluated many times using, for each evaluation, component reliabilities drawn at random from their given distributions. These overall reliability values are accumulated in the form of a histogram which is then taken to be an approximation to the required distribution.

The following tabulations give the results of the Monte Carlo assessment for the reliability of Structures 1 and 2, using 5000 simulated system values.

Reliability histogram for Structure 1 :-

reliability range	frequency
0.00 to 0.05	0
0.05 to 0.10	0
0.10 to 0.15	0
0.15 to 0.20	0
0.20 to 0.25	0
0.25 to 0.30	0
0.30 to 0.35	0
0.35 to 0.40	0
0.40 to 0.45	0
0.45 to 0.50	0
0.50 to 0.55	0
0.55 to 0.60	0
0.60 to 0.65	1
0.65 to 0.70	0
0.70 to 0.75	9
0.75 to 0.80	62
0.80 to 0.85	260
0.85 to 0.90	903
0.90 to 0.95	2070
0.95 to 1.00	1695

Mean reliability = 0.9253
Variance estimate = 0.001949

Reliability histogram for Structure 2 :-

reliability range	frequency
0.00 to 0.05	0
0.05 to 0.10	0
0.10 to 0.15	0
0.15 to 0.20	0
0.20 to 0.25	0
0.25 to 0.30	0
0.30 to 0.35	0
0.35 to 0.40	0
0.40 to 0.45	0
0.45 to 0.50	0
0.50 to 0.55	0
0.55 to 0.60	0
0.60 to 0.65	0
0.65 to 0.70	1
0.70 to 0.75	0
0.75 to 0.80	3
0.80 to 0.85	66
0.85 to 0.90	398
0.90 to 0.95	1739
0.95 to 1.00	2793

Mean reliability = 0.9474
Variance estimate = 0.001117

B.4 Program listing

The following is the Pascal program for the Monte Carlo assessment of control system reliability.

```
program controlsystems (input, output);

{     This program reads data which represents a network of components
with a start component numbered zero, a finish component numbered N
intermediate components numbered 1..N-1, and one way links between the
components forming various routes from component zero to component N.
A list of all possible routes is built up and then an expression is
formed which represents the reliability of the whole network in terms of
the reliabilities of the individual components.

     The program requests values for the Beta or Gamma parameters of
each node and then uses a Monte Carlo technique to evaluate the
reliability of the network count ( = 5000 ) times using a pseudo-random
generator to assign reliability values to the nodes, with each
reliability being derived from the appropriate Beta or Gamma
```

```
distribution.    A  histogram is assembled for the  overall  reliability
distribution.  }

const
  max = 50;       { maximum component number }
  count = 5000;   { number of values for the overall
                    reliability histogram }

type
  range = 0..max;
  collection = set of range;

  ptr = ^node;
  node = record
           path : collection;
           coeff, lastin : integer;
           next : ptr
         end;

  point = ^family;
  family = record
           father, size : range;
           sons : collection;
           kin : point
           end;

  params = array [range,1..2] of real;

var
  route, q : ptr;
  result, mean, varn : real;
  reliability : array [range] of real;
  source, sink :range;
  links : point;
  betapars : params;
  x, y, z, i, k, nof0 :integer;
  nodelist : array [range] of -1..1;

{ a node list element = -1 if the node has not yet been mentioned
                      =  0 if the node has been given as an input
                      =  1 if the node outputs have been supplied  }

  histo : array [0..19] of integer; { Array to hold the results of }
                                    { a Monte Carlo simulation for }
                                    { reliability as a histogram.  }

  a : array [1..32] of real;    { The arrays a, t, h, d are required  }
  t,h : array [1..31] of real;  { for the generation of variates from }
  d : array [6..47] of real;    { the normal or Gamma distributions.  }
```

```
function random:real;

{ This random number generator is based upon N.P.L. Report DITC 6/82 by
B.A.Wichmann and I.D.Hill.  The global variables x, y, z are to be
given initial random integer values which should be less than 30000.  }

  var
    w, ran : real;

  begin
    repeat
      x := 171*(x mod 177) - 2*(x div 177);
      if x<0 then x := x + 30269;
      y := 172*(y mod 176) - 35*(y div 176);
      if y<0 then y := y + 30307;
      z := 170*(z mod 178) - 63*(z div 178);
      if z<0 then z := z + 30323;
      w := x/30269 + y/30307 + z/30323;
      ran := w - trunc(w)
   until ran <> 0;
   random := ran
 end;

{ The following procedures FL GM GS GT GO are from Computing vol.12,
pages 223 to 246 (1974) by J.H.Ahrens and U.Dieter.  They are
efficient routines for computing random variates from a normal
distribution (FL) and from a gamma distribution (GM GS GT GO).  The
function betagamma returns a random reliability either as a Beta variate
or as exp( - Gamma variate).  }

procedure fl(var x : real);

  label
    1, 4, 6, 13, 14, 17;

  var
    u, us, tt, w, y, aa : real;
    s, i : integer;

  begin
1:  u := random;
    s := trunc(2*u);
    u := trunc(32*(2*u - s));
    i := trunc(u);
    if i <> 0 then
    begin
      us := u - i;
      aa := a[i];
4:    if us > t[i] then
      begin
        w := (us - t[i])*h[i];
        goto 17
      end;
```

```
      u := random;
      w := u*(a[i+1] - aa);
      tt := (w/2 + aa)*w;
6:    if us > tt then goto 17;
      u := random;
      if us < u then
      begin
        us := random;
        goto 4
      end;
      tt := u;
      us := random;
      goto 6
    end
    else
    begin
      i := 6;
      aa := a[32];
      u := 2*u;
      while u < 1.0 do
      begin
        aa := aa + d[i];
        i := i + 1;
        if i > 47 then goto 1;
        u := 2*u
      end;
      u := u - 1;
13:   w := u*d[i];
      tt := (w/2 + aa)*w;
14:   us := random;
      if us > tt then goto 17;
      u := random;
      if us < u then
      begin
        u := random;
        goto 13
      end;
      tt := u;
      goto 14
    end;
17: y := aa + w;
    if s = 0 then x := y else x := -y
  end;
```

```
procedure gm(n : integer; var x : real);

  var
    p : real;
    i : integer;

  begin
    p := 1;
    for i := 1 to n do
    p := p*random;
    x := - ln(p)
  end;


procedure gs(a : real; var x : real);

  label 1;

  var
    p, b, u, xx : real;

  begin
    b := (2.718281828459 + a)/2.718281828459;
1:  u := random;
    p := b*u;
    if p > 1 then
    begin
      xx := - ln((b - p)/a);
      if random > exp((a - 1)*ln(xx)) then goto 1 else x := xx
    end
    else
    begin
      xx := exp(ln(p)/a);
      if random > exp(-xx) then goto 1 else x := xx
    end
  end;


procedure gt(a : real; var x : real);

  var
    m : integer;
    f, z, y :real;

  begin
    m := trunc(a);
    f := a - m;
    if m = 0 then y := 0 else gm(m,y);
    if f = 0 then z := 0 else gs(f,z);
    x := y + z
  end;
```

```
procedure go(a : real; var x : real);

  label
    2, 7, 6, 8, 10;

  var
    mu, v, sig, sig2, w, d, b, u, s, ss, xx :real;

  begin
    mu := a - 1;
    v := sqrt(a);
    sig2 := a + 1.632993161855*v;
    sig := sqrt(sig2);
    w := sig2/mu;
    d := 2.44948974278318*sig;
    b := mu + d;
2:  u := random;
    if u <= 0.009572265238289 then goto 8;
    fl(s);
    xx := mu + sig*s;
    if (xx < 0) or (xx > b) then goto 2;
    u := random;
    ss :=sqr(s)/2;
    if s >= 0 then goto 6;
    if u <= 1 - ss*((1 - 2*s/v)*w -1) then goto 10 else goto 7;
6:  if u <= 1 - ss*(w-1) then goto 10;
7:  if ln(u) > mu*(1 + ln(xx/mu)) - xx + ss then goto 2 else goto 10;
8:  s := - ln(1 - random);
    xx := b*(1 + s/d);
    u := random;
    if ln(u) > mu*(2 + ln(xx/mu) - xx/b)+3.7203284924588 - b -ln(sig*d/b)
      then goto 2;
10: x := xx
  end;
```

```
function betagamma(a, b : real):real;

{ Procedure to compute a reliability from the random number generator.
A beta variate is returned for the reliability if a, b are positive with
the variate proportional to the cumulative distribution of x**(a-1) *
(1 - x)**(b-1).  If a, b have been entered as negative values then this
indicates that the required reliability is to be derived from a Gamma
distribution and the value returned is exp(-y/|b|), where y is a Gamma
variate from the cumulative distribution of x**(|a| - 1) * exp(-x).
The value y/|b| is the corresponding variate from the cumulative
distribution for the expression x**(|a| - 1) * exp(-|b|*x). }

  var
    x, y, c :real;

  begin
    c := abs(a);
    if c <= 3 then gt(c,x) else go(c,x);
    if a < 0 then betagamma := exp( -x/abs(b))
    else
    begin
      if b <= 3 then gt(b,y) else go(b,y);
      betagamma := x/(x+y)
    end
  end;


procedure split (var route, hd : ptr;  n : range);

{ Split route into two lists,  one called hd containing those  records
for  which n is a member of their path field,  the other  list  called
route contains all the other records. }

  var
    p : ptr;

  begin
    if route <> nil then
    if n in route^.path then
    begin
      p := route;
      route := p^.next;
      p^.next := hd;
      hd := p;
      split (route, hd, n)
    end
    else split (route^.next, hd, n)
  end;
```

```
procedure display (s : collection);

  var
    i : range;

  begin
    for i := 0 to max do
    if i in s then write (i : 1, ' ');
    writeln
  end;


procedure show (route : ptr);

  begin
    while route <> nil do
    with route^ do
    begin
      write (coeff : 5, ' : ');
      write (lastin : 5, ' : ');
      display (path);
      route := next
    end
  end;


procedure create (var route : ptr);

{  Read in all the data for the network and create  the  corresponding
linked list called route. }

  var
    i, j, m, n : range;
    p, r : ptr;
    kith : point;
    s : collection;
    done, test : boolean;

  begin

{ first find all connections in kith giving the father node,  the  set
  of sons of the father and the number size of these sons }

    repeat
      n := 0;
      while nodelist[n] <> 0 do n := n + 1;
      nodelist[n] := 1;
      nof0 := nof0 - 1;
      writeln ('Number of connections from node ', n : 1, ' ? ');
      readln (i);
      new(kith);
      kith^.father := n;
      kith^.size := i;
      kith^.sons := [];
```

```
      if i = 0 then sink := n;
      if (n <> source) and (n <> sink) then
      begin
        writeln('Reliability parameters of node ',n : 1,' ? ');
        writeln('Enter the negative values for Gamma distribution');
        readln(betapars[n,1],betapars[n,2])
      end;
      if i > 0 then
      begin
        for j := 1 to i do
        begin
          writeln ('Connection ', j : 1, ' from node ', n : 1, ' ? ');
          readln (m);
          kith^.sons := kith^.sons + [m];
          if nodelist[m] = -1 then
          begin
            nodelist[m] := 0;
            nof0 := nof0 + 1
          end
        end
      end;
      kith^.kin := links;
      links := kith
    until nof0 = 0;

{ now use the connections in links to find all pathways through the
  graph from source to sink }

    new(route);
    with route^ do
    begin
      path := [0];
      coeff := 1;
      lastin := 0;
      next := nil
    end;
    repeat
      p := route;
      done := true;
      repeat
        if p <> nil then
        if (sink in p^.path) or (p^.lastin < 0) then p := p^.next;
        test := p = nil;
        if not test then
          test := not ( sink in p^.path) and ( p^.lastin >= 0)
      until test;
      if p <> nil then
      if not ( sink in p^.path) and (p^.lastin >= 0) then
      begin
        kith := links;
        while kith^.father <> p^.lastin do kith := kith^.kin;
        s := kith^.sons;
        for j := 1 to kith^.size do
```

```
        if j < kith^.size then
        begin
          m := 0;
          while not ( m in s ) do m := m + 1;
          s := s - [m];
          new(r);
          r^ := p^;
          r^.path := r^.path + [m];
          if not ( m in p^.path) then
          begin
            r^.lastin := m;
            done := false
          end
          else r^.lastin := -m;
          r^.next := route;
          route := r
        end
        else
        begin
          m := 0;
          while not ( m in s ) do m := m + 1;
          if not ( m in p^.path ) then
          begin
            done := false;
            p^.path := p^.path + [m];
            p^.lastin := m
          end
          else p^.lastin := - m
        end
      end
    until ( p  = nil ) and done
  end;
```

```
procedure comb (hd : ptr;  var route : ptr);

{ Convert the linked list of paths into the linked list of reliability
  terms. }

  var
    p, q, r : ptr;
    s : collection;
    done : boolean;

  begin
    p := hd;
    q := route;
    while q <> nil do
    with q^ do
    begin
      new (p^.next);
      p := p^.next;
      p^.path := path + hd^.path;
      p^.coeff := - coeff * hd^.coeff;
      p^.lastin := sink;
      q := next
    end;
    p^.next := nil;
    while hd <> nil do
    begin
      s := hd^.path;
      q := route;
      done := false;
      repeat
        if q^.path = s then
        begin
          q^.coeff := q^.coeff + hd^.coeff;
          if q^.coeff = 0 then
          begin
            if q = route then route := q^.next
            else r^.next := q^.next;
            dispose (q)
          end;
          done := true
        end
        else
        begin
          r := q;
          q := q^.next
        end
      until done or (q = nil);
      if done then
      begin
        r:=hd;
        hd := hd^.next;
        dispose (r)
      end
      else
```

```
      begin
        r^.next := hd;
        hd := hd^.next;
        r^.next^.next := nil
      end
    end
  end;


procedure combine (route : ptr);

  var
    p, hd : ptr;

  begin
    hd := route^.next;
    route^.next := nil;
    while hd <> nil do
    begin
      p := hd^.next;
      comb (hd, route);
      hd := p
    end
  end;
```

```
procedure evaluate (route : ptr;  var result : real);

{ Compute the numerical expression for the overall reliability. }

  var
    p : ptr;
    x : real;
    s : collection;
    i : integer;

  begin
    result := 0.0;
    p := route;
    while p <> nil do
    begin
      s := p^.path;
      x := p^.coeff;
      i := 0;
      while s <> [] do
      begin
        if i in s then
        begin
          x := x * reliability[i];
          s := s - [i]
        end;
        i := i + 1;
      end;
      result := result + x;
      p := p^.next
    end
  end;


begin
nodelist[0] := 0;
for i := 1 to max do nodelist[i] := -1;
nof0 := 1;
a[1] := 0;
a[2] := 0.03917608550309; a[3] := 0.07841241273311;
a[4] := 0.11776987457909; a[5] := 0.15731068461017;
a[6] := 0.19709908429430; a[7] := 0.23720210932878;
a[8] := 0.27769043982157; a[9] := 0.31863936396437;
a[10] := 0.36012989178957; a[11] := 0.40225006532172;
a[12] := 0.44509652498551; a[13] := 0.48877641111466;
a[14] := 0.53340970624127; a[15] := 0.57913216225555;
a[16] := 0.62609901234641; a[17] := 0.67448975019607;
a[18] := 0.72451438349236; a[19] := 0.77642176114792;
a[20] := 0.83051087820539; a[21] := 0.88714655901887;
a[22] := 0.94678175630104; a[23] := 1.00999016924958;
a[24] := 1.07751556704027; a[25] := 1.15034938037600;
a[26] := 1.22985875921658; a[27] := 1.31801089730353;
a[28] := 1.41779713799625; a[29] := 1.53412054435253;
a[30] := 1.67593972277344; a[31] := 1.86273186742164;
a[32] := 2.15387469406144;
```

```
for i := 1 to 31 do
   begin
   t[i] := ((a[i+1] - a[i])/2 + a[i])*(a[i+1] - a[i]);
   h[i] := (a[i+1] - a[i])/(1 - t[i])
   end;
d[6] := 0.26368432217502; d[7] := 0.24250845238097;
d[8] := 0.22556744380930; d[9] := 0.21163416577204;
d[10] := 0.19992426749317; d[11] := 0.18991075842246;
d[12] := 0.18122518100691; d[13] := 0.17360140038056;
d[14] := 0.16684190866667; d[15] := 0.16079672918053;
d[16] := 0.15534971747692; d[17] := 0.15040938382813;
d[18] := 0.14590257684509; d[19] := 0.14177003276856;
d[20] := 0.13796317369537; d[21] := 0.13444176150074;
d[22] := 0.13117215026483; d[23] := 0.12812596512583;
d[24] := 0.12527909006226; d[25] := 0.12261088288608;
d[26] := 0.12010355965651; d[27] := 0.11774170701949;
d[28] := 0.11551189226063; d[29] := 0.11340234879117;
d[30] := 0.11140272044119; d[31] := 0.10950385201710;
d[32] := 0.10769761656476; d[33] := 0.10597677198479;
d[34] := 0.10433484129317; d[35] := 0.10276601206127;
d[36] := 0.10126505151402; d[37] := 0.09982723448906;
d[38] := 0.09844828202068; d[39] := 0.09712430874765;
d[40] := 0.09585177768776; d[41] := 0.09462746119186;
d[42] := 0.09344840710526; d[43] := 0.09231190933664;
d[44] := 0.09121548217294; d[45] := 0.09015683778986;
d[46] := 0.08913386650005; d[47] := 0.08814461935364;
x := 31415;
y := 27188;
writeln('Input random seed, an integer between 0 and 30000');
readln(z);
source := 0;
sink := max;
links := nil;
create (route);
q := route;
combine(route);
show(route); writeln; writeln;
reliability[0] := 1.0;
reliability[sink] := 1.0;
for i := 0 to 19 do histo[i] := 0;
mean := 0.0;
varn := 0.0;
for i := 1 to count do
begin
  for k := 1 to sink-1 do
  reliability[k] := betagamma(betapars[k,1],betapars[k,2]);
  evaluate (route, result);
  mean := mean + result;
  varn := varn + sqr(result);
  k := trunc(20*result);
  histo[k] := histo[k] + 1
end;
writeln('      range          frequency');
```

```
for i := 0 to 19 do
writeln(i*0.05:7:2,' to ',(i+1)*0.05:4:2,histo[i]);
mean := mean/count;
varn := (varn - count * sqr(mean))/(count - 1);
writeln; writeln('   Mean reliability =',mean);
writeln('   Variance estimate =',varn)
end.
```

B.5 Procedures to compute beta and gamma functions

The following Pascal procedures are appended in case numerical values are required for Beta or Gamma functions. They do not form part of the program for analysing the control structures.

```
function logama( x : real):real;

{ compute log gamma(x) function }

  var
    xc, f, z :real;

  begin
    xc := x;
    if xc >= 7.0 then f := 0.0
    else
    begin
      f := 1.0;
      z := xc;
      while z < 7.0 do
      begin
        xc := z;
        f := f*z;
        z := z + 1.0
      end;
      xc := xc + 1.0;
      f := - ln(f)
    end;
    z := 1.0/sqr(xc);
    logama := f + (xc-0.5)*ln(xc) - xc + 0.918938533 +
        (((4.0*z-3.0*z)*z-14.0)*z+420.0)/(5040.0*xc)
  end;

function logbeta( p, q:real):real;

{ compute complete log beta function }

  begin
    logbeta := logama(p) + logama(q) - logama(p+q)
  end;
```

```
function betain( x, p, q, zeta : real; var ifault : integer):real;

{ compute incomplete beta function ratio for argument x between
0 and 1 and for p, q positive }

  label
    3, 4, 5;

  const
    accuracy = 1.0e-8;

  var
    index ,ok : boolean;
    psq, cx, pp, qq, xx, term, ai, rx, temp, beta : real;
    ns : integer;

  begin
    beta := x;
    ifault := 0;
    if (p <= 0.0) or (q <= 0.0) then ifault := 1;
    if (x < 0.0) or (x > 1.0) then ifault := 2;
    ok := ifault = 0;
    if ok and (x <> 0.0) and (x <> 1.0) then
    begin
      psq := p + q;
      cx := 1.0 - x;
      index := p < psq*x;
      if index then
      begin
        xx := cx;
        cx := x;
        pp := q;
        qq := p
      end
      else
      begin
        xx := x;
        pp := p;
        qq := q
      end;
      term := 1.0;
      ai := 1.0;
      beta := 1.0;
      ns := trunc(qq + cx*psq);
      rx := xx/cx;
3:    temp :=qq - ai;
      if ns = 0 then rx := xx;
4:    term := term*temp*rx/(pp+ai);
      beta := beta + term;
      temp := abs(term);
      if (temp <= accuracy) and (temp <= accuracy*beta) then goto 5;
      ai := ai + 1.0;
      ns := ns - 1;
      if ns >= 0 then goto 3;
```

```
      temp :=psq;
      psq := psq + 1.0;
      goto 4;
5:    beta := beta*exp(pp*ln(xx)+(qq-1.0)*ln(cx) - zeta)/pp;
      if index then beta := 1.0 - beta
    end;
    betain := beta
  end;

function betafn( x, p, q :real):real;

{ compute incomplete beta function ratio, see change below }
{ for how to compute the incomplete beta function itself.  }

  var
    a, r : real;
    j : integer;

  begin
    r := logbeta(p,q);
    a := betain(x,p,q,r,j);
    if j = 0 then betafn := a  { for beta function replace := a }
                               { with := a*exp(r)                }
    else
    begin
      writeln('Error in incomplete beta function');
      writeln('x=',x:4,'  should be between 0 and 1');
      writeln('p=',p:4,'  should be > 0');
      writeln('q=',q:4,'  should be > 0')
    end
  end;
```

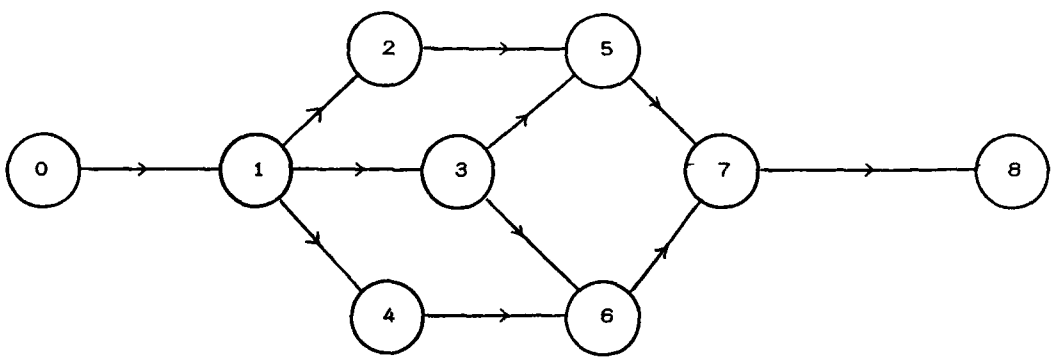


Fig. 1

END

DATE

FILMED

8 88